Case Study : Disease prediction with Symptoms Analysis

By :

Pandarge Nikhil N 1504012

Bhusnar Pradip S 1404037

Guided By:

Prof.Rankhambe J P

Code:

```
#---------------------------

# Disease prediction with Symptoms analysis

# @author Nikhil N Pandarge

# @author Pradip S Bhusnar

#-------------------------
```

code:

```
getwd()

bc_data <- read.table("breast-cancer-wisconsin.data",
                      header = FALSE,
                      sep = ",")
head(bc_data)

colnames(bc_data) <- c("sample_code_number",
```

```
                "clump_thickness",
                "uniformity_of_cell_size",
                "uniformity_of_cell_shape",
                "marginal_adhesion",
                "single_epithelial_cell_size",
                "bare_nuclei",
                "bland_chromatin",
                "normal_nucleoli",
                "mitosis",
                "classes")

head(bc_data)
```

Output:

```
> head(bc_data)
  sample_code_number clump_thickness uniformity_of_cell_size
uniformity_of_cell_shape
1            1000025               5                       1
                      1
2            1002945               5                       4
                      4
3            1015425               3                       1
                      1
4            1016277               6                       8
                      8
5            1017023               4                       1
                      1
6            1017122               8                      10
                     10
  marginal_adhesion single_epithelial_cell_size bare_nuclei
bland_chromatin normal_nucleoli
1                 1                           2           1
              3              1
2                 5                           7          10
              3              2
3                 1                           2           2
              3              1
4                 1                           3           4
              3              7
5                 3                           2           1
              3              1
6                 8                           7          10
              9              7
```

	mitosis	classes
1	1	2
2	1	2
3	1	2
4	1	2
5	1	2
6	1	4

Code:

```
bc_data$classes <- ifelse(bc_data$classes == "2", "benign",
                          ifelse(bc_data$classes == "4", "malignant", NA))

bc_data[bc_data == "?"] <- NA

#how many NAs are in the data

length(which(is.na(bc_data)))
```

Output:

16

Code:

```
# how many samples would we loose, if we removed them?

nrow(bc_data)
```

Output:

699

Code:

```
nrow(bc_data[is.na(bc_data), ])
```

Output:

16

Code:

```
#-------------------------
# impute missing data

install.packages("mice")

library(mice)

bc_data[,2:10] <- apply(bc_data[, 2:10], 2, function(x)
as.numeric(as.character(x)))

dataset_impute <- mice(bc_data[, 2:10],  print = FALSE)

bc_data <- cbind(bc_data[, 11, drop = FALSE],

mice::complete(dataset_impute, 1))

bc_data$classes <- as.factor(bc_data$classes)

# How many benign and malignant cases are there?

summary(bc_data$classes)
```

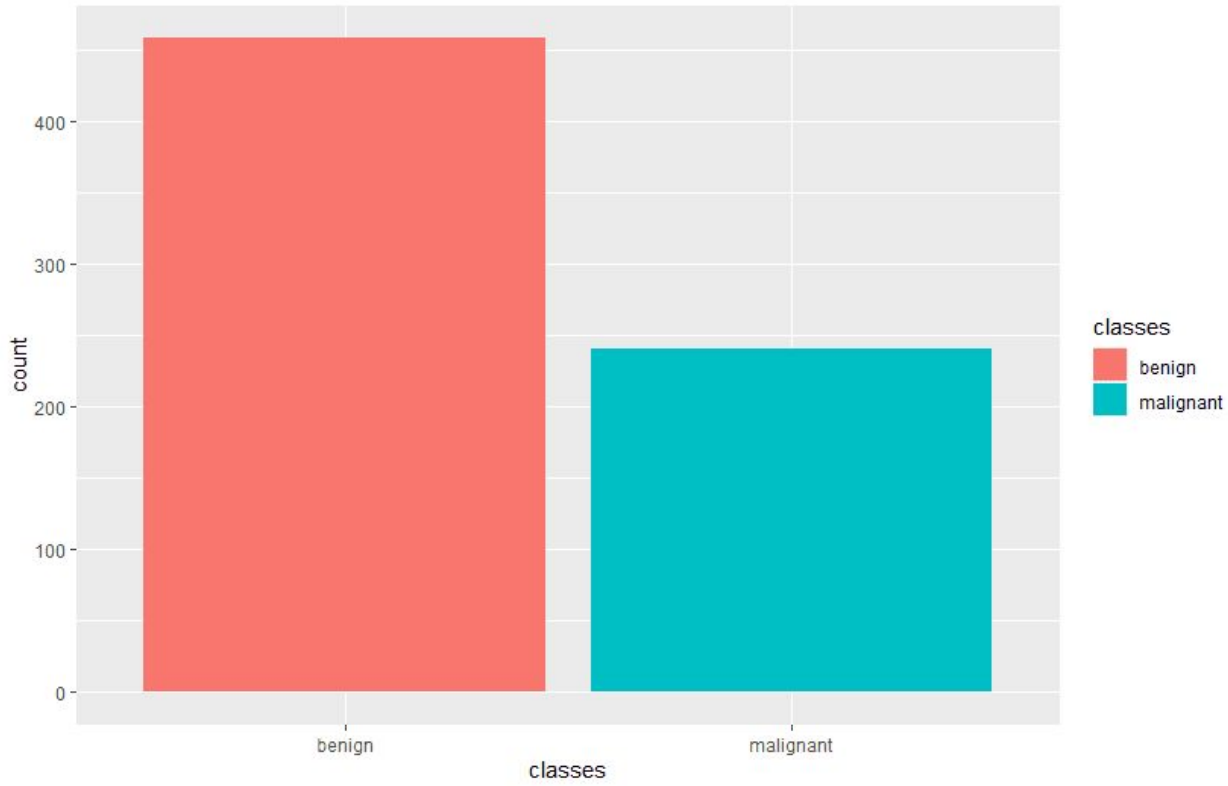
Output:

```
> summary(bc_data$classes)
   benign malignant
      458       241
```

Code:

```
library(ggplot2)

ggplot(bc_data, aes(x = classes, fill = classes)) +
  geom_bar()
```

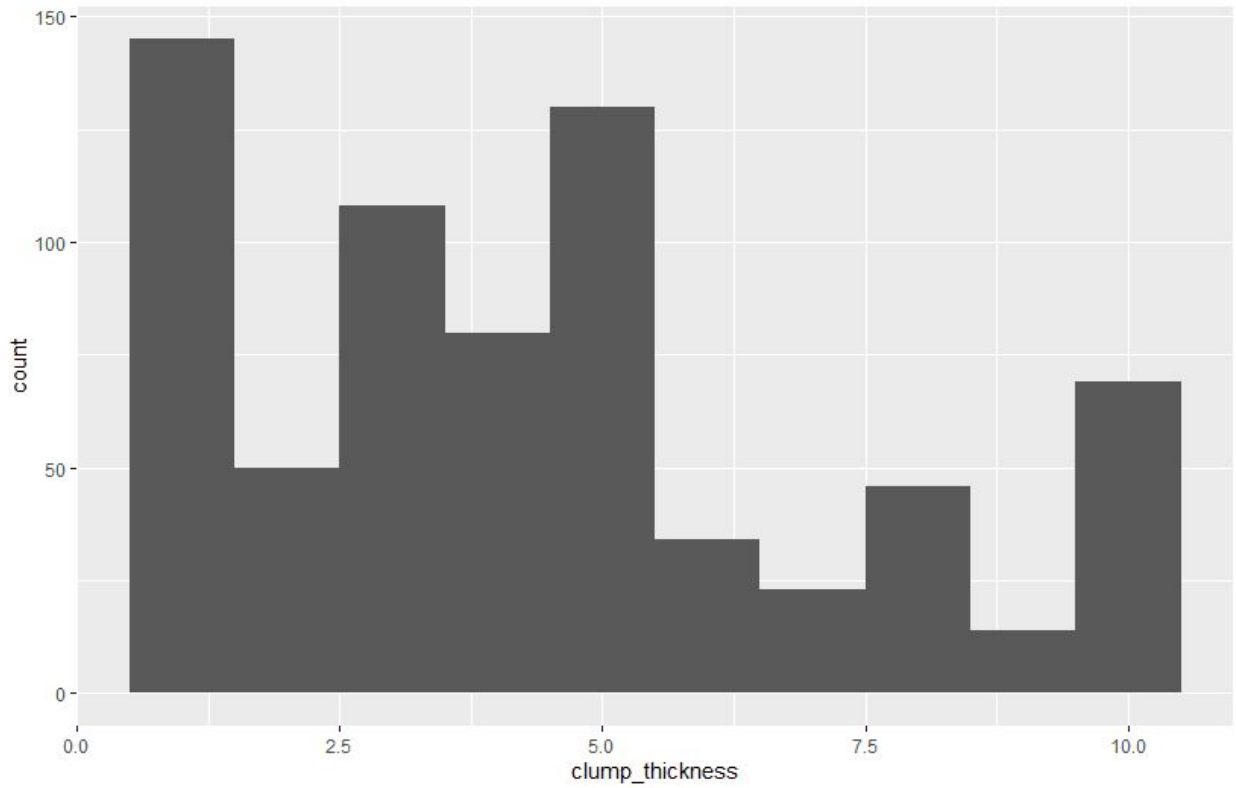
Output:



Code:

```
ggplot(bc_data, aes(x = clump_thickness)) +
  geom_histogram(bins = 10)
```

Output:

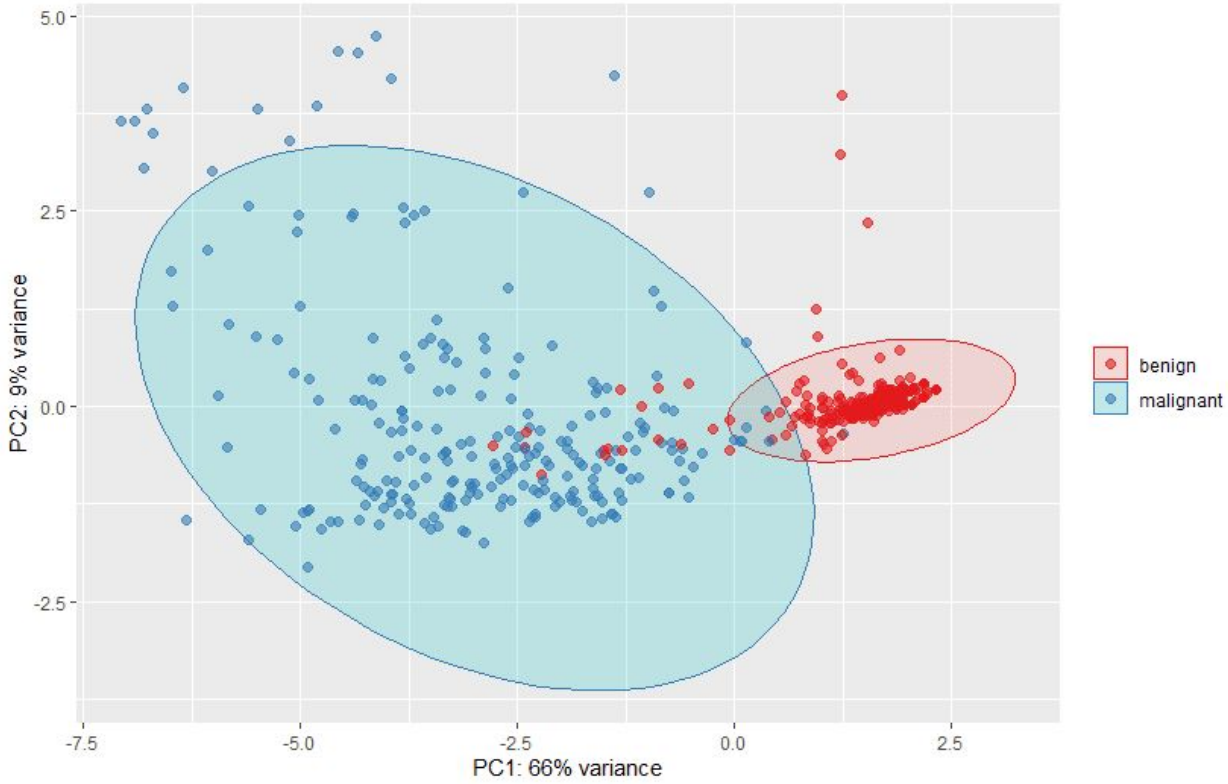


Code:

```
# principal component analysis:

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("pcaGoPromoter")

library(pcaGoPromoter)

library(ellipse)

# perform pca and extract scores

pcaOutput <- pca(t(bc_data[, -1]), printDropped = FALSE, scale = TRUE,
center = TRUE)

pcaOutput2 <- as.data.frame(pcaOutput$scores)

# define groups for plotting

pcaOutput2$groups <- bc_data$classes

centroids <- aggregate(cbind(PC1, PC2) ~ groups, pcaOutput2, mean)

conf.rgn  <- do.call(rbind, lapply(unique(pcaOutput2$groups), function(t)
  data.frame(groups = as.character(t),
             ellipse(cov(pcaOutput2[pcaOutput2$groups == t, 1:2]),
                     centre = as.matrix(centroids[centroids$groups == t, 2:3]),
                     level = 0.95),
             stringsAsFactors = FALSE)))
```

```
ggplot(data = pcaOutput2, aes(x = PC1, y = PC2, group = groups, color =
groups)) +
  geom_polygon(data = conf.rgn, aes(fill = groups), alpha = 0.2) +
  geom_point(size = 2, alpha = 0.6) +
  scale_color_brewer(palette = "Set1") +
  labs(color = "",
       fill = "",
       x = paste0("PC1: ", round(pcaOutput$pov[1], digits = 2) * 100, "%
variance"),
       y = paste0("PC2: ", round(pcaOutput$pov[2], digits = 2) * 100, "%
variance"))
```

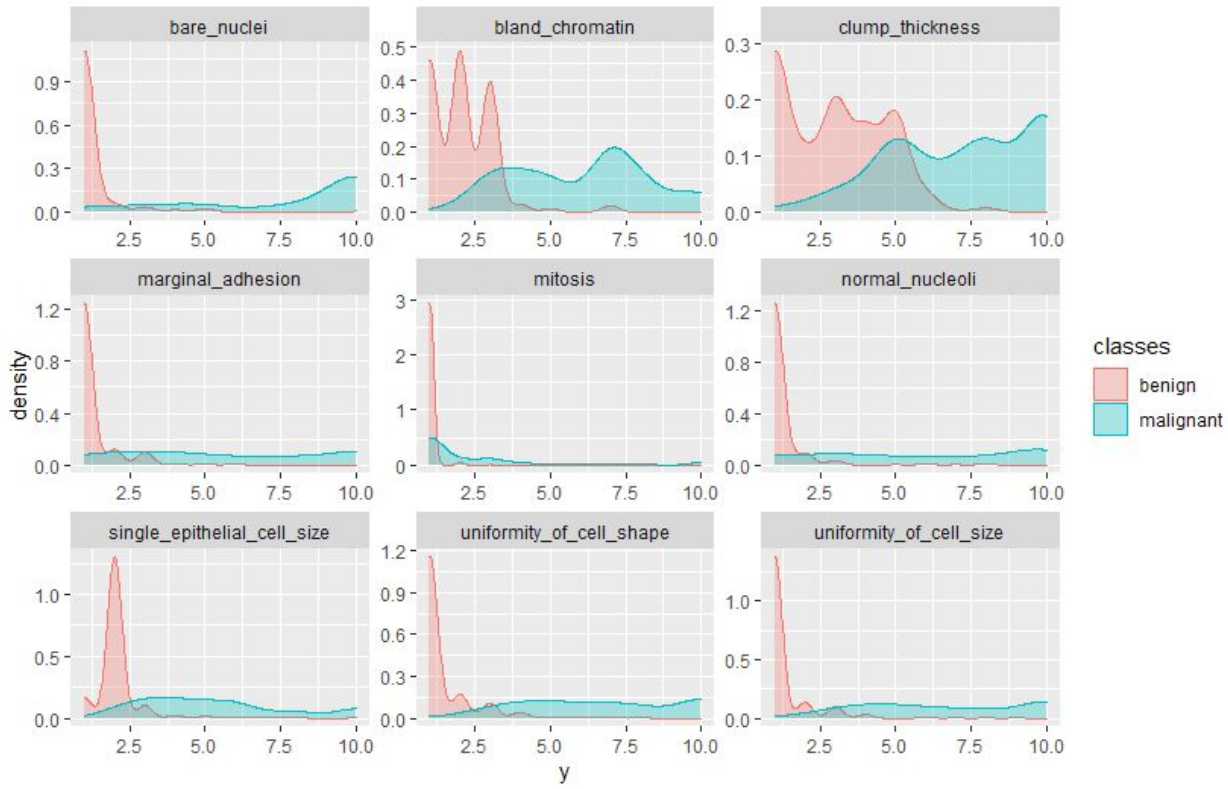
Output:



code:

```
library(tidyr)

gather(bc_data, x, y, clump_thickness:mitosis) %>%
  ggplot(aes(x = y, color = classes, fill = classes)) +
  geom_density(alpha = 0.3) +
  facet_wrap( ~ x, scales = "free", ncol = 3)
```

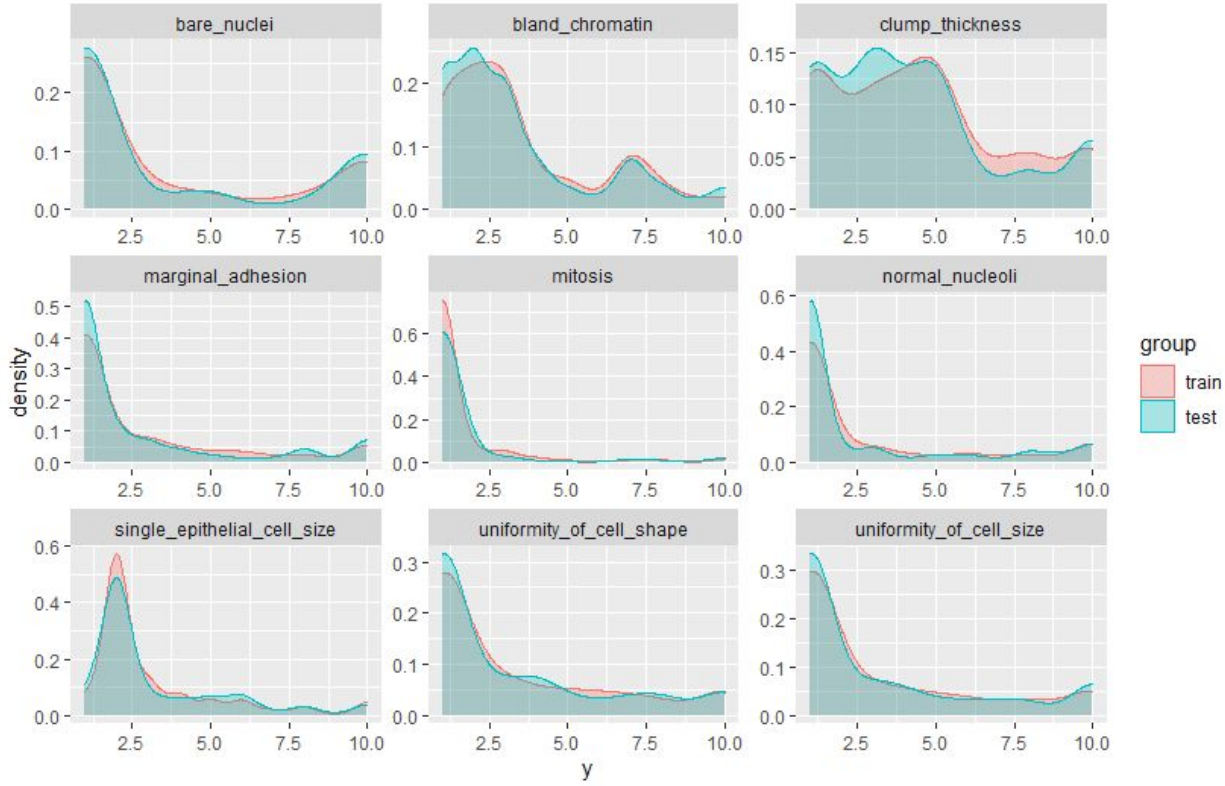
Output:



Code:

```
# configure multicore
install.packages("doParallel")


library(doParallel)

cl <- makeCluster(detectCores())
registerDoParallel(cl)

library(caret)

#Training, validation and test data

set.seed(42)
index <- createDataPartition(bc_data$classes, p = 0.7, list = FALSE)
train_data <- bc_data[index, ]
test_data  <- bc_data[-index, ]

library(dplyr)

rbind(data.frame(group = "train", train_data),
      data.frame(group = "test", test_data)) %>%
  gather(x, y, clump_thickness:mitosis) %>%
  ggplot(aes(x = y, color = group, fill = group)) +
  geom_density(alpha = 0.3) +
  facet_wrap( ~ x, scales = "free", ncol = 3)
```

Output:



Code:

```
set.seed(42)
model_glm <- caret::train(clump_thickness ~ .,
                          data = train_data,
                          method = "glm",
                          preProcess = c("scale", "center"),
                          trControl = trainControl(method = "repeatedcv",
                                                   number = 10,
                                                   repeats = 10,
                                                   savePredictions = TRUE,
                                                   verboseIter = FALSE))
model_glm
```

```
predictions <- predict(model_glm, test_data)
```

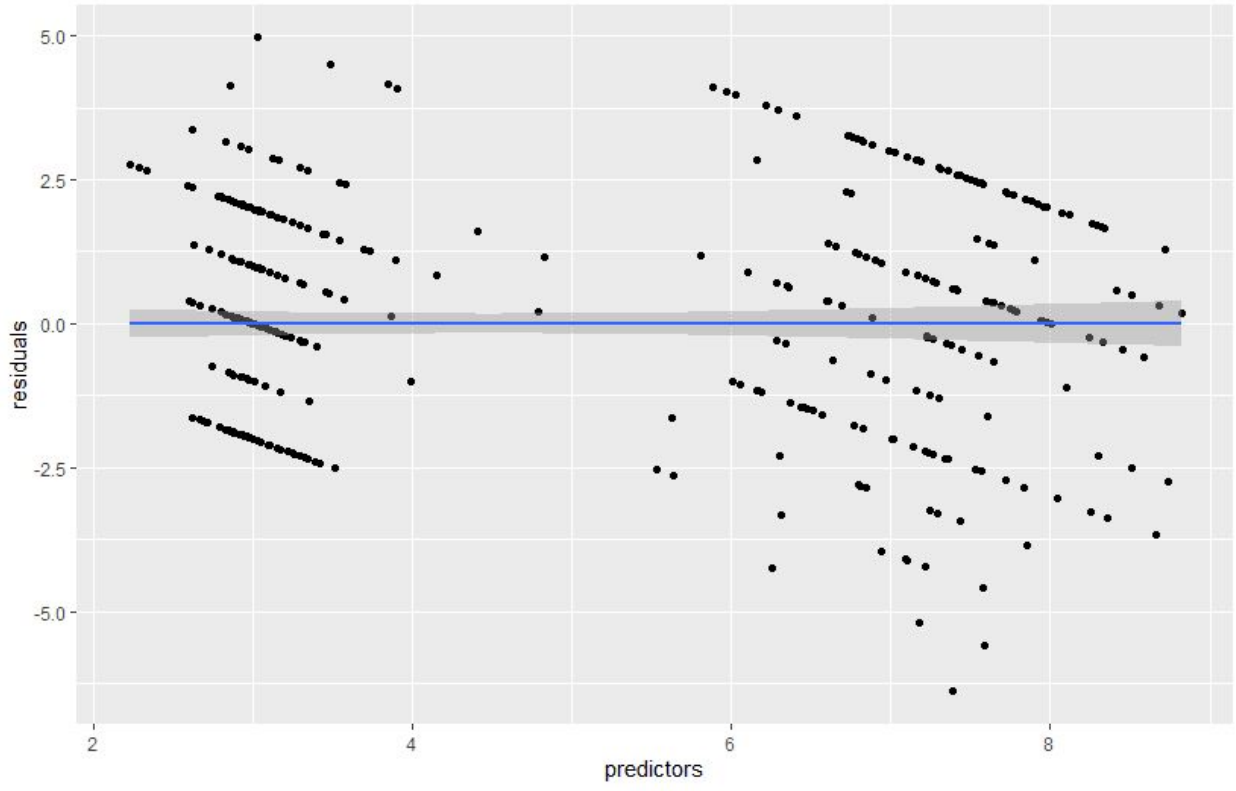
Output:

```
> model_glm
Generalized Linear Model

490 samples
  9 predictor

Pre-processing: scaled (9), centered (9)
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 442, 440, 443, 442, 441, 441, ...
Resampling results:

  RMSE      Rsquared   MAE
  1.951781  0.5273127  1.62851
```

Code:

```
predictions <- predict(model_glm, test_data)

# model_glm$finalModel$linear.predictors ==
model_glm$finalModel$fitted.values
 # residual is the difference between observed value and predicted value.

data.frame(residuals = resid(model_glm),
           predictors = model_glm$finalModel$linear.predictors) %>%
  ggplot(aes(x = predictors, y = residuals)) +
  geom_jitter() +
  geom_smooth(method = "lm")
```

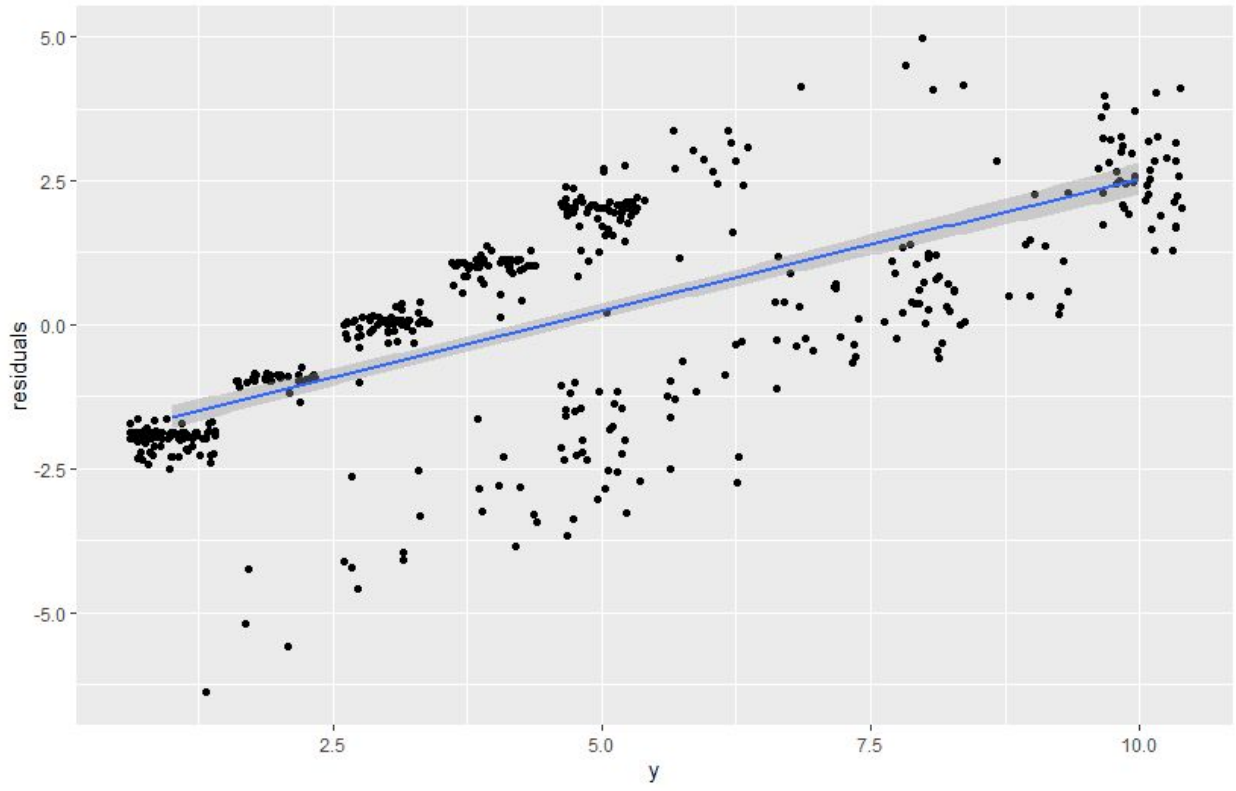
Output:



Code:

```
# y == train_data$clump_thickness:

data.frame(residuals = resid(model_glm),
           y = model_glm$finalModel$y) %>%
  ggplot(aes(x = y, y = residuals)) +
  geom_jitter() +
  geom_smooth(method = "lm")
```

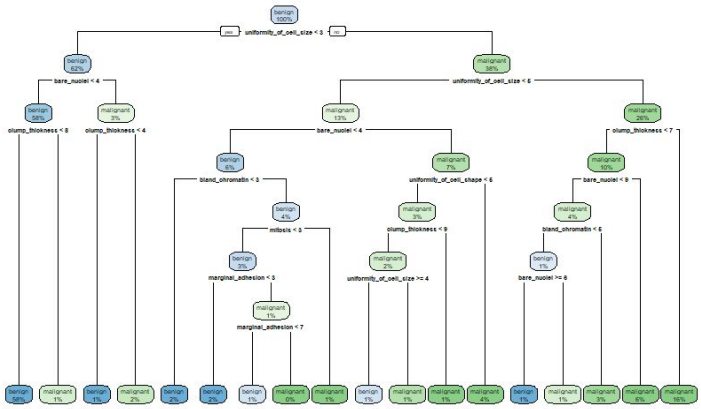
Output:



Code:

```
#Classification:

library(rpart)

library(rpart.plot)

set.seed(42)

fit <- rpart(classes ~ .,
             data = train_data,
             method = "class",
             control = rpart.control(xval = 10,
                                     minbucket = 2,
                                     cp = 0),
             parms = list(split = "information"))

rpart.plot(fit, extra = 100)
```

Output:



Code:

```
#rain forest

set.seed(42)
model_rf <- caret::train(classes ~ .,
                         data = train_data,
                         method = "rf",
                         preProcess = c("scale", "center"),
                         trControl = trainControl(method = "repeatedcv",
                                                  number = 10,
                                                  repeats = 10,
                                                  savePredictions = TRUE,
                                                  verboseIter = FALSE))


model_rf$finalModel$confusion


imp <- model_rf$finalModel$importance
imp[order(imp, decreasing = TRUE), ]
```

Output:

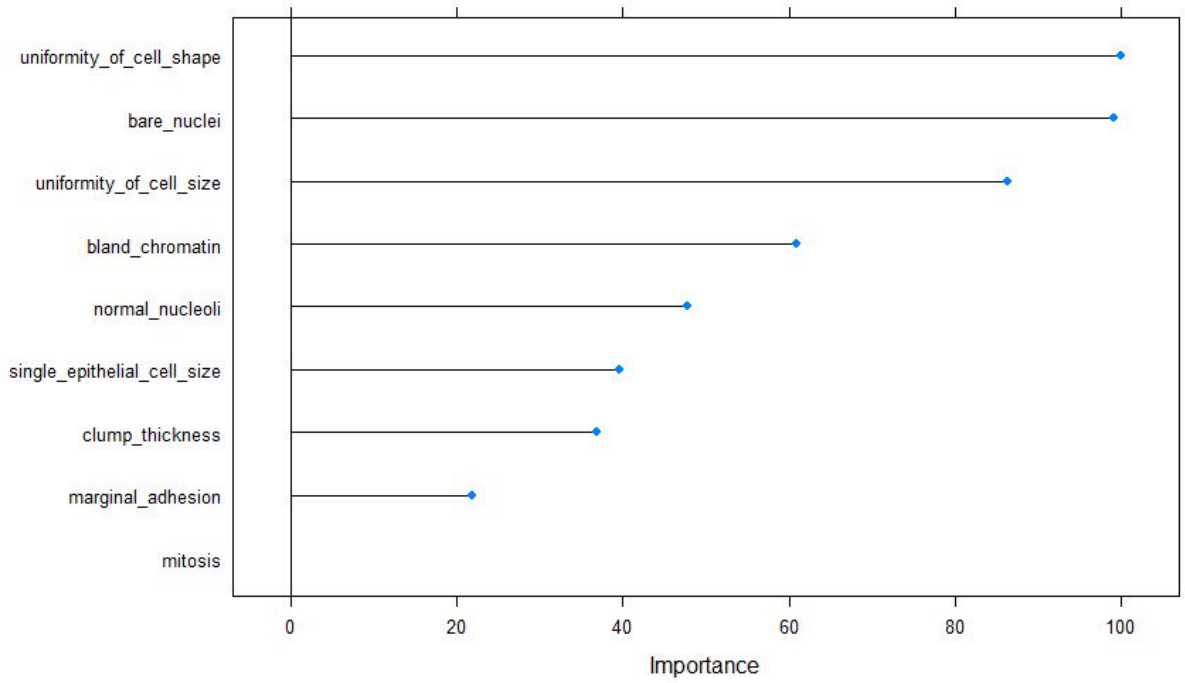
```
> model_rf$finalModel$confusion
          benign malignant class.error
benign       310        11  0.03426791
malignant      6       163  0.03550296
> imp <- model_rf$finalModel$importance
> imp[order(imp, decreasing = TRUE), ]
               bare_nuclei     uniformity_of_cell_size
uniformity_of_cell_shape
```

43.079030 38.388266 36.212991
bland_chromatin normal_nucleoli
single_epithelial_cell_size
28.917296 20.936824 19.624624
clump_thickness marginal_adhesion mitosis
17.791695 11.687941 2.961946

Code:

```
# estimate variable importance

importance <- varImp(model_rf, scale = TRUE)

plot(importance)
```

Output:



Code:

```
confusionMatrix(predict(model_rf, test_data), test_data$classes)
```

Output:

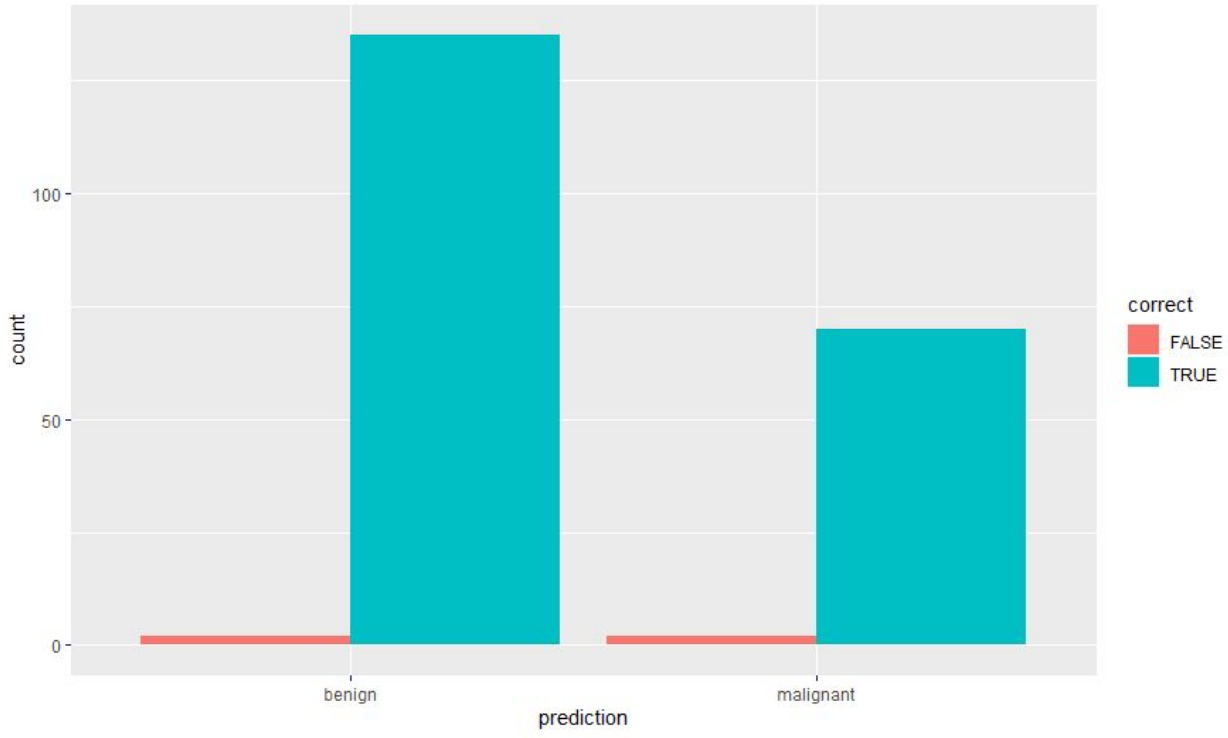
```
> confusionMatrix(predict(model_rf, test_data), test_data$classes)
Confusion Matrix and Statistics

          Reference
Prediction  benign malignant
  benign       135         2
  malignant      2        70

               Accuracy : 0.9809
                 95% CI : (0.9517, 0.9948)
    No Information Rate : 0.6555
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9576

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9854
            Specificity : 0.9722
         Pos Pred Value : 0.9854
         Neg Pred Value : 0.9722
             Prevalence : 0.6555
         Detection Rate : 0.6459
   Detection Prevalence : 0.6555
      Balanced Accuracy : 0.9788
```

'Positive' Class : benign

Code:

```
results <- data.frame(actual = test_data$classes,
                      predict(model_rf, test_data, type = "prob"))

results$prediction <- ifelse(results$benign > 0.5, "benign",
                             ifelse(results$malignant > 0.5, "malignant", NA))

results$correct <- ifelse(results$actual == results$prediction, TRUE,
FALSE)

ggplot(results, aes(x = prediction, fill = correct)) +
  geom_bar(position = "dodge")
```

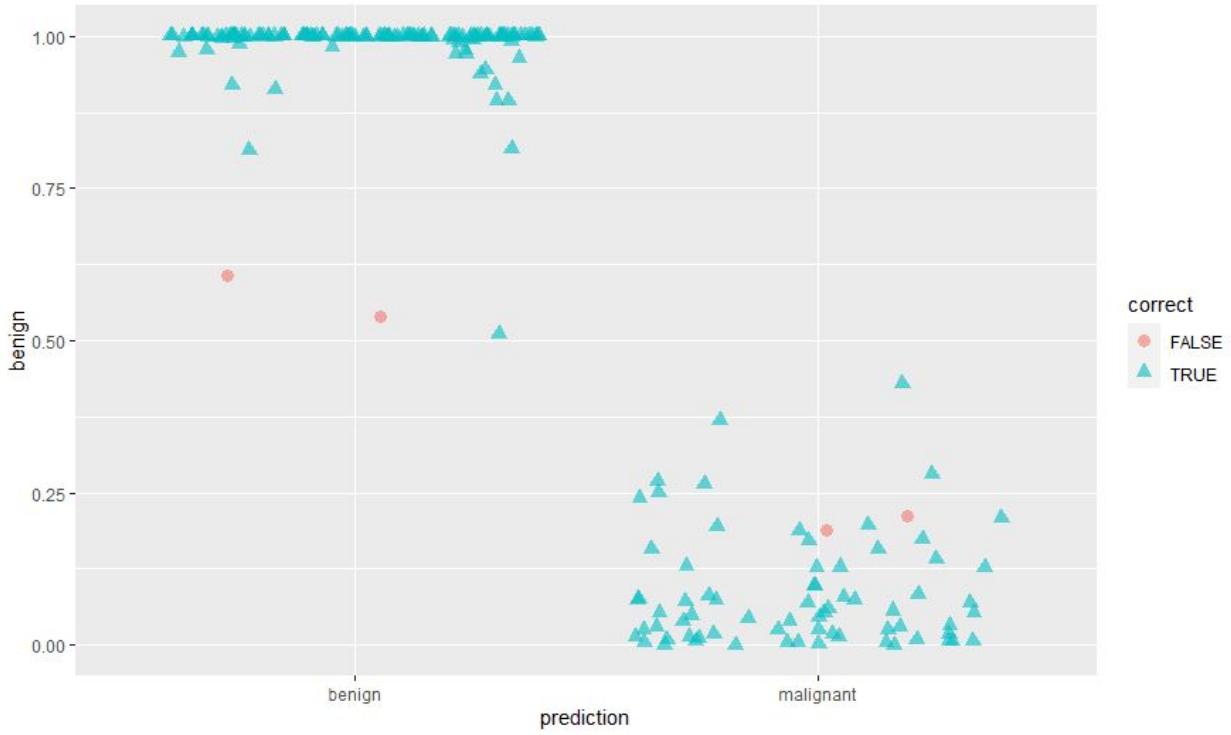
Output:



Code:

```
ggplot(results, aes(x = prediction, y = benign, color = correct, shape =
correct)) +
  geom_jitter(size = 3, alpha = 0.6)
```

Output:



Code:

```
#Feature Selection:

library(corrplot)

# calculate correlation matrix
corMatMy <- cor(train_data[, -1])
corrplot(corMatMy, order = "hclust")
```

Output: